COMP 5413
Topics in Smart Health Informatics
Assignment 3

Name: Khushal Paresh Thaker
Student ID: 1106937

The dataset contains CT images of Abdomen, Chest and Head. The images are resized to 32*32 and split into separate folders.

Preprocessing:

ImageDataGenerator class from the keras.preprocessing library is used to perform the necessary preprocessing on the images. The images are *normalized* by setting the rescale argument to 1/255 so that each pixel which when multiplied with, achieves a range between 0 and 1. *rotation_range* is set to 40, a value within which the images are randomly rotated. *width_shift_range and height_shift_range* randomly translates the images vertically and horizontally for a range of 0.2. As there are no assumptions on the horizontal asymmetry, *horizontal_flip* is assigned true to randomly flip half the images horizontally. *fill_mode* allows newly created pixels to be filled, which can appear after a rotation or a width/height shift.

A CNN model is created to extract the features from 3 categories of CT images – Abdomen, Chest and Head. The model contains Conv 2d layer, max pooling, Batch Normalisation, Dense and also Flatten layer. Dropout is added to reduce overfitting. The model is run with Adam as the optimizer with varying learning rates (0.1, 0.01, 0.001) in order to avoid over-fitting and under fitting as the learning rate. Categorical cross entropy as the loss function used as the parameters are one-hot encoded and normalized. The best model is saved and later loaded to evaluate the test parameters as well. The best accuracy is 67.43%% acquired for learning rate – 0.001. Although, the model that ran with learning rate 0.01 had an accuracy of 68%, validation accuracy was 35% and this showed a huge difference which could be because of overfitting. The validation accuracy for the model with learning rate 0.001 reached 67.11% with fluctuations between 35.78%. This model had a stable accuracy of 66% as shown with learning rate 0.001 as shown in figure 3. The training loss settles at about 0.48 while the validation reduces drastically and then stabilises at 49 as shown in figure 3.

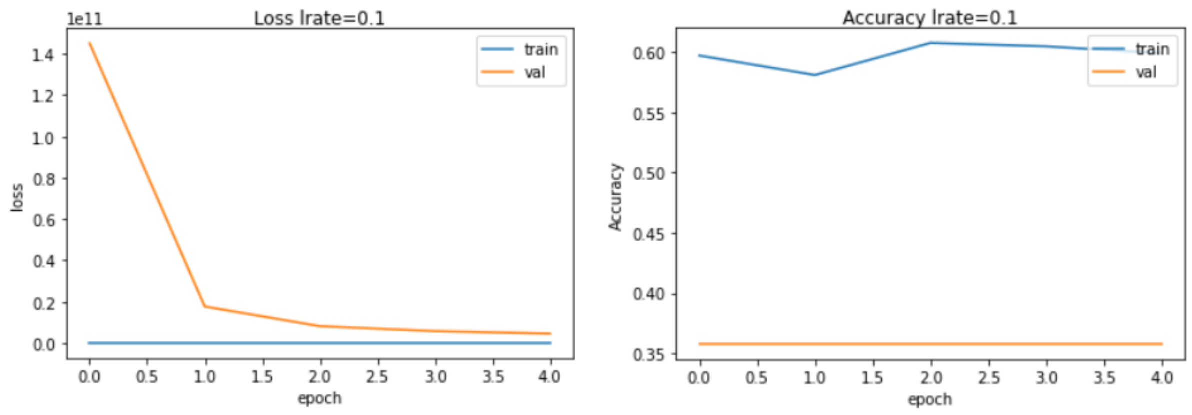

Fig 1: Loss and Accuracy for CNN (lr = 0.1)

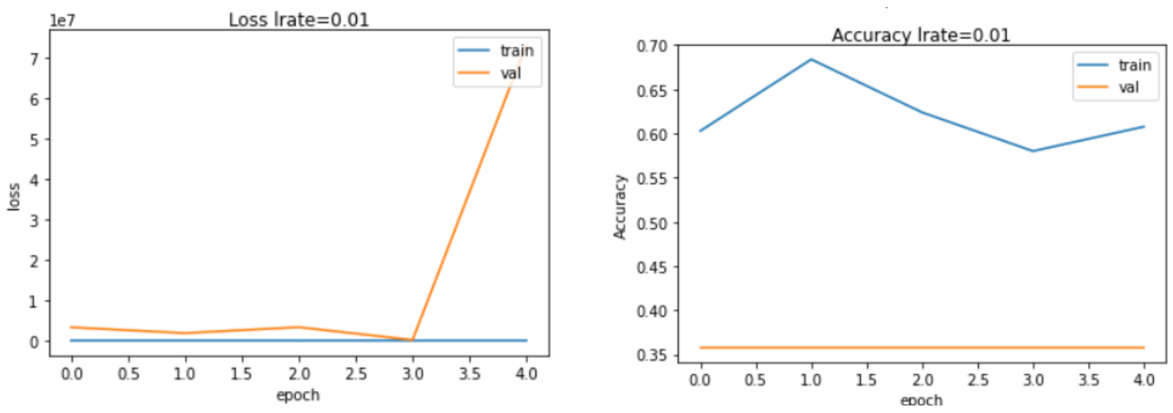

Fig 2: Loss and Accuracy for CNN (lr = 0.01)

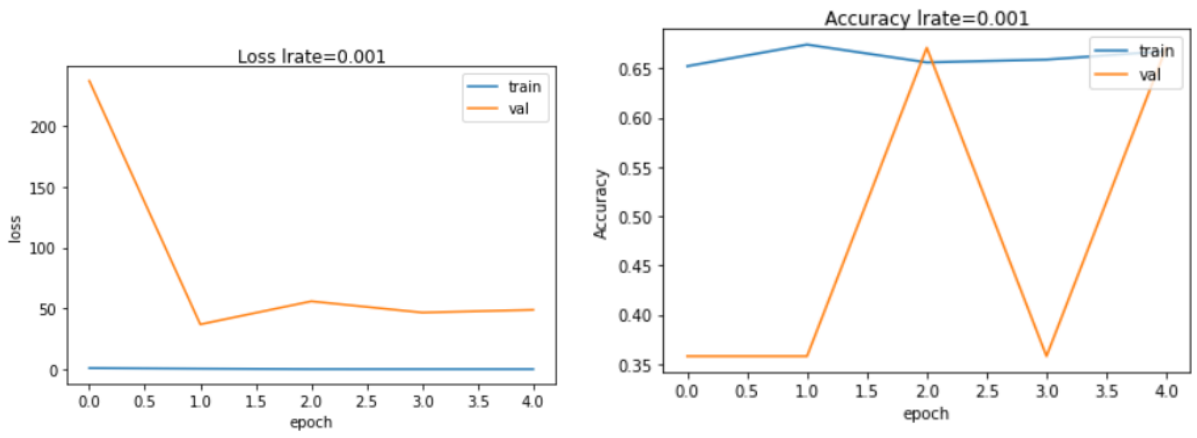

Fig 3: Loss and Accuracy for CNN (lr = 0.001)

The accuracy of the model while testing reaches 67.11%

For learning rate 0.1, model seems to be stagnant on validation accuracy around 32% which is very less. The training accuracy is a lot more than that at 60% which could indicate overfitting. The training loss remains stagnant while validation loss reduces drastically as shown in figure 1. For 0.01 as the learning rate, the accuracy reaches 68% in 2nd epoch but then fluctuates and reaches 60%, where as the validation accuracy is stagnant at 35% which shows massive overfitting. This could be because of oversampling as there are only 3 categories and 150 images for each split between training and testing. Moreover, it takes a lot of time to tune the model.

With learning rate as 0.001, Accuracy for validation and training accuracy remains similar. Training accuracy being 66.7%. As seen in the Figure 3, Accuracy for training fluctuates between 65% and 45%. The validation accuracy stabilises after the 2nd epoch at 67.11% as well. Also, training loss is 0.48, reducing gradually which would be better to classify the images. The model could be suffering from over fitting as the validation and training accuracy stagnates after the 2nd epoch. Although, Drop out layers have been added to reduce the extent of overfitting.

Features are extracted from the intermediate layer (Layer32)

Classifiers:

KNN

The features are extracted using the CNN model and the inputs are passed on to the KNN with K ranging in [3,5,7,9]

Accuracy is 76.19%, highest at k =3.

While for other k values it is,

K = 5, accuracy is 74.09%

K = 7, accuracy is 72.38%

K = 9, accuracy is 72.28%

K = 3, Accuracy is 76.09%

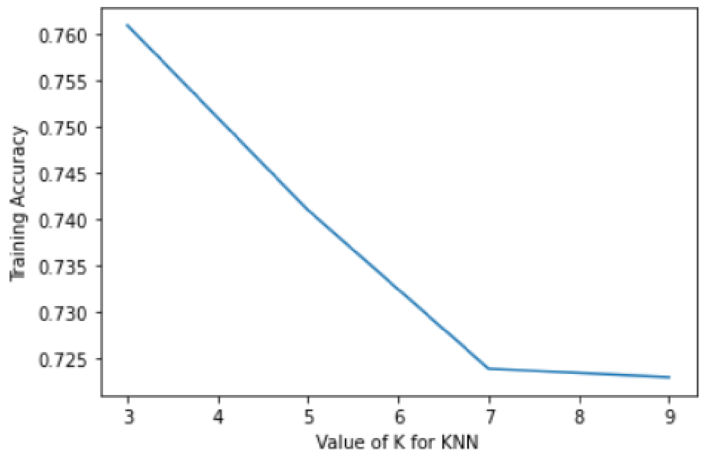

Fig 4: Accuracy vs K for KNN

This gradual decrease in accuracy could be because of over fitting.

When measured using the recall score,

Confusion Matrix:

256	71	25
27	326	6
86	76	177

	Precision	Recall	F1
Abdomen	0.69	0.73	0.71
Chest	0.69	0.91	0.78
Head	0.85	0.52	0.65

Table 1: Precision, Recall, F1 Score for KNN (Training)

Once the saved pickle model is loaded and evaluated against the test values, Accuracy is 69.56% considering the model saved the best k (= 3).

Confusion Matrix:

95	39	14
13	128	0
37	34	90

	Precision	Recall	F1
Abdomen	0.66	0.64	0.65
Chest	0.64	0.91	0.75
Head	0.87	0.56	0.68

Table 2: Precision, Recall, F1 Score for KNN (Testing)

The testing doesn't show any massive change, with a little reduction when compared to the evaluation done with training parameters.

The accuracy for KNN reduces with increasing K value which could be because of overfitting as with increasing k, the data gets smoother. Also, with less data for each category, overfitting is predominant. F1 score is around 68% as well.

The testing accuracy is around 69% showing consistency with the training data.

Random Forest

Similar to KNN, the features are extracted separately for train and test.

The Random forest model is run and the below report is generated for testing after the model is save and evaluated with test parameters, the report for F1 score is,

Confusion Matrix:

73	34	41
20	101	20
47	34	80

	Precision	Recall	F1
Abdomen	0.52	0.49	0.51
Chest	0.60	0.72	0.65
Head	0.57	0.50	0.53

Table 3: Precision, Recall, F1 Score for RF (Testing) – Before Hyperparameter Tuning and Confusion Matrix

Accuracy is 56%.

To optimise the model, random search is used to tune certain hyper parameters.

Max_depth, min_sample_leaf, boothstrap, criterion and n_estimators are the hyper parameters used to tune the model.

Max depth is taken so as to control the depth of the tree growth to 3. Increasing it a lot, might reduce the accuracy. Min_sample_leaf allows us to control the growth of the tree between 1 to 11 randomly and allows in preventing overfitting. N_estimators are nothing but the number of trees required to train. Increasing the number of trees results in higher accuracy. Although, the accuracy might stagnate after a certain point, it doesn't drop down.

We iterate it for 20 providing a random state of 42. We use k-cross validation (k=4) to improve the results.

Once, the model is run, we get the following best parameters:

Bootstrap: True, *Criterion*: Entropy, *max_depth*: 3, *min_sample_leaf*: 8, *n_estimators*: 617

The model provides the below F1 score:

Confusion Matrix:

236	112	4
2	356	1
103	94	142

	Precision	Recall	F1
Abdomen	0.69	0.67	0.68
Chest	0.63	0.99	0.77
Head	0.97	0.42	0.58

Table 4: Precision, Recall, F1 Score for RF (Training) – Before Hyperparameter Tuning and Confusion Matrix

And the Accuracy is found to be 70%

The model is saved and run with the test parameters, achieving:

Confusion Matrix:

88	60	0
1	140	0
61	41	59

	Precision	Recall	F1
Abdomen	0.59	0.59	0.59
Chest	0.58	0.99	0.73
Head	1.0	0.37	0.54

Table 5: Precision, Recall, F1 Score for RF (Testing) – After Hyperparameter Tuning and Confusion Matrix

The testing accuracy of 63.78% after tuning is more than the model before tuning. This means, that better hyperparameters were chosen to tune the model. Also, the testing accuracy provided by the F1 score is 63.78% which is not far behind the training accuracy at 69.9%. Having chosen the depth accordingly, makes sure that the model is not overfitting as one of the reason for overfitting in random forest could be high depth. Also, cross validation (k = 4) reduces the variance estimate of the model's true sample score. Figure 5 represents a heat map of the random forest after tuning with hyperparameters. The accuracy increases with shade getting lighter for every criteria's and estimators.

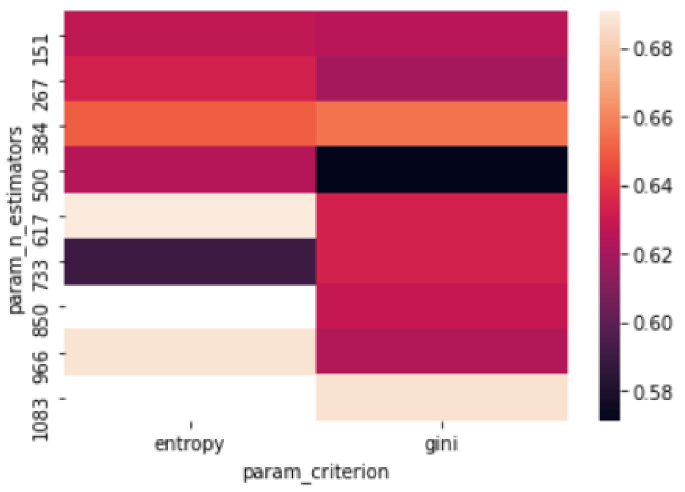

Fig 5: Heat map for Random forest with

Random Search for the two criterions: entropy and gini